First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
End of Result Set

□ Generate Collection Print

L11: Entry 3 of 3 File: USPT Oct 12, 1999

DOCUMENT-IDENTIFIER: US 5966704 A
TITLE: Storage plane organization and storage systems based thereon using queries and subqueries for data searching

Brief Summary Text (6):
The vast majority of the mentioned higher software layers for the handling of externally stored data are either commercially available as products, or are part of an operating system and provided together with it, or are available from the public software domain. In principle, they apply to nearly all of the presently installed and known computing systems, covering likewise, to give some examples, high-end parallel supercomputers, large business computing systems, commercial and engineering midrange systems, and workstations down to simple personal computers. On all these platforms, storage devices with direct access capability form the backbone of most modern applications for the processing of externally stored data. Consequently, data handling software with the capability to exploit such direct access storage devices is a key component in the overall design and implementation of modern applications. Data are stored and accessed by application programs through the use of the following two main categories of data handling software:

Drawing Description Text (2):
The invention is described in detail below with reference to the following schematic drawings:

Drawing Description Text (9):
FIG. 6 is a schematic illustration of a storage system, according to the present invention; and

Drawing Description Text (10):
FIGS. 7A-7C are schematic illustrations of the basic elements of a storage system, according to the present invention.

Detailed Description Text (2):
In FIG. 1, a schematic representation of a logical storage plane 10, according to the present invention, is given. This storage plane 10 is a two-dimensional plane which may grow without limitations. The plane is structured in columns corresponding to individual types of information units (type-columns) extending parallel to the y-axis. These type-columns are subdivided into storage segments 11 indicated as rectangular boxes of different size. (It is to be noted that some or all of these storage segments may have the same size). The storage segments 11 can be dealt with as containers each having a certain storage capacity and configuration, also referred to as storage segment structure.

Detailed Description Text (35):
In the following, an implementation of a storage system, according to the present invention, is described as first embodiment. This storage system is schematically illustrated in FIG. 6. From a physical point of view, such a storage system comprises one or several computers, each of them with their attached workstations, terminals, and storage devices, all being locally attached or interconnected by means of a network. The network can either be a wireless or a conventional one. Company wide computer systems usually consist of interconnected heterogeneous computing platforms of different processor types, numerous kind of storage devices, and a large number and variety of terminals and workstations. In FIG. 6, such a heterogeneously composed storage system is shown. There are different application programs 51 and terminals or terminal emulators 52 that have access to the storage devices 57, 58 and 59. These storage devices can be accessed through File, Database or Document Management Systems such as, for example, IBM's DB2 (box 53), an Item Access Facility (IAFC) (box 54) used in connection with an application

running on IBM's Customer Information Control System (CICS), VisualInfo (box 55), or KODAK ImageLink (box 56). In a conventional system, each application program or user would have to deal with all the different File, Database or Document Management Systems directly. In addition to the fact that the application program or user has to understand and know how to deal with each of the File or Database or Document Systems, it is important to know prior to searching or retrieving a particular information unit where this information unit is actually stored. According to the present invention, all these and other disadvantages are overcome by providing a master index 60 such that one has direct access to all relevant index sections of all the information units stored in the different storage devices and locations.

Detailed Description Text (44):
In the following, an example of a storage system, its basic elements, and its functions will be described in connection with FIGS. 7A-7C. In the present example, a request for searching 701 is produced by a terminal 700. This request is delivered to the storage system 702. This system 702 comprises means 703 for determining the respective information unit types which are implicitly or explicitly addressed or defined by said request for searching 701, i.e., certain information unit type columns 704, 705 and 706 are identified in a logical storage plane 707. This means 703 are followed by means 708 for analyzing the request for searching 701 in order to determine the time frame concerned for each such information unit type columns 704, 705 and 706. In the present example it is schematically illustrated that a time frame is assigned to each of said information unit type columns. Next, means 709 are employed for determining which storage segment or segments 710-715 are affected by said request for searching 701, i.e., which storage segments are selected by means of said time frames in each of said information unit type columns 704, 705 and 706. Means 716 for identifying the storage medium or media 718-723 on which said storage segment or segments 710-715 reside are now employed. This means 716 helps to select those storage media A1, B18, B19, B20, C1, C2 (media 718-723) from all storage media A1, A2, B5-B25, C1, C2, C43 forming part of said storage system 702 which are to be taken into account. In a next step, means 717 for generating a specific set of data manipulation language statements (DMLS) are employed. These means 717 provide executable search subrequests for each of said storage media 718-723, for each version of the storage segments 710-715 in said time frame, the different versions of storage segments being determined based on pre-established mapping relations 724 describing the mapping of said information unit types 704, 705, 706 of a certain version into a storage segment type of a certain version. Note that the generation of a particular DMLS may be carried out in several cascading steps. For example, if data are to be retrieved from storage subsystems such as IAFC 54 or KODAK ImageLink 56, DMLS are generated on behalf of the particular storage subsystem which in turn generates its own specific DMLS to access its storage media 718, 723. Said executable search subrequests are now carried out by means 725, i.e., the respective data manipulation language statement (DMLS) is delivered to each storage medium 718-723. From the index portions of these storage media 718-723 reference pointers Hc2, Hc1, Hb20, Hb19, Hb18, Ha1 are retrieved and means 726 for consolidating a hit list 727 are employed. This hit list 727 is basically a list comprising reference pointers to the body data (and annotation data) not shown in FIG. 7C. Either a subset of hits in said hit list 727, or all hits can now be used to retrieve the respective body and annotation data (not shown in FIG. 7C). In the present example, only two reference pointers, namely Hc1 and Ha1 were selected. This means that only the storage media 721 and 718 are now addressed. A building block 728, referred to as means for retrieving information units, is now activated to retrieve the body data. The reference pointer Ha1 points to three rows 729-731 of data, whereas pointer Hc1 only points to two rows 732 and 733 of data.

Current US Original Classification (1):
707/3

Current US Cross Reference Classification (1):
707/4

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

□ Generate Collection Print

L11: Entry 1 of 3 File: USPT Oct 28, 2003

DOCUMENT-IDENTIFIER: US 6640278 B1
TITLE: Method for configuration and management of storage resources in a storage network

Drawing Description Text (18):
FIG. 17 illustrates schematically a persistent storage hardware driver module according to the present invention.

Detailed Description Text (113):
Accordingly, within the storage platform, hardware functions (such as disk or flash storage) and software functions (such as RAID stripes or Mirrors) are all accessed via software drivers most commonly referred to as devices.

Detailed Description Text (208):
The storage domain manager offers a comprehensive set of centralized storage management capabilities that can be leveraged from a single management interface, across all attached servers and storage, regardless of vendor. From a central location, a system administrator may control the movement or mirroring of data between heterogeneous storage resources, and can dynamically leverage these capabilities across different heterogeneous storage resources over time. This results in a significant cost savings and simplification of administrative complexity. As a scalable, intelligent platform, the storage domain manager resides in the perfect central location to host storage management functionality that can be leveraged across all attached server and storage resources.

Current US Cross Reference Classification (1):
707/1

Current US Cross Reference Classification (2):
707/10

Current US Cross Reference Classification (3):
707/102

Previous Doc Next Doc Go to Doc#

L14: Entry 2 of 7 File: USPT Oct 28, 2003

DOCUMENT-IDENTIFIER: US 6640278 B1
TITLE: Method for configuration and management of storage resources in a storage network

Abstract Text (1):
A storage domain management system supports storage domains. The storage server includes a plurality of communication interfaces. A first set of communication interfaces in the plurality is adapted for connection to all kinds of users of data. A second set of communication interfaces in the plurality is adapted for connection to respective devices in a pool of storage devices for use in a storage domain. Data processing resources in the server are coupled to the plurality of communication interfaces for transferring data among the interfaces. The data processing resources comprise a plurality of driver modules and configurable logic linking driver modules into data paths. Each configured data path acts as a virtual circuit that includes a set of driver modules selected from the plurality of driver modules. A data storage transaction which is received at a communication interface is mapped to one of the configured data paths. A display and a user input device are included with data processing structures to manage images displayed on the display.

Parent Case Text (3):
The present invention is related to U.S. Pat. No. 6,538,669, entitled Graphical User Interface for Configuration of a Storage System, invented by Richard A. Lagueux, Jr., Joel H. Stave, John B. Yeaman, Brian E. Stevens, Robert M. Higgins, and James M. Collins, issued Mar. 25, 2003, which patent is owned by Dell Products L.P.

Brief Summary Text (11):
The present invention provides a system for managing storage resources in a storage network according to storage domains. The system includes a plurality of communication interfaces, adapted for connection via communication media to clients and storage systems and the storage network. A processing unit is coupled with the plurality of communication interfaces and includes logic to configure a set of storage locations from the one or more storage systems in the network as a storage domain for a set of at least one client from the one or more clients in the storage network. The system includes in various combinations elements providing multi-protocol support across the plurality of communication interfaces, logic to route storage transactions within a storage domain in response to the transaction identifiers carried within the protocols, a management interface for configuring the storage domains, logic for translating a storage transaction traversing the plurality of communication interfaces into and out of a common format for routing within the system among the plurality of communication interfaces, resources for caching the data subject of storage transactions, and logic to manage the migration of data sets from one storage location to another storage location within the network.

Brief Summary Text (14):
In a preferred embodiment, the resources within storage domains are defined using virtual circuits which comprise a plurality of driver modules and configurable logic linking driver modules into data paths, which are implemented in pairs for redundancy in a preferred system. Each configured data path acts as a virtual circuit that includes a set of driver modules selected from the plurality of driver modules. A data storage transaction which is received at a communication interface is mapped to one of the configured data paths, and thereby controlled within a storage domain managed and configured in the storage domain manager.

Drawing Description Text (9):
FIG. 8 is a simplified diagram of a hardware driver module for a fibre channel interface for use in the system of the present invention.

Drawing Description Text (19):
FIG. 18 illustrates an example opening image for a graphical user interface according to the present invention.

Detailed Description Text (4):
The storage server 1200 in the network has client interfaces 1210, 1211, 1212 coupled to client servers 1201, 1202, and 1203, respectively. Storage interfaces 1213 and 1214 are coupled via communication channels to storage devices 1205, 1206, 1207 which, when combined with any storage in the storage server 1200, provide physical storage for a storage domain managed in the storage server 1200. The communication channel 1213 in this example is connected through a hub 1204 to the devices 1205 and 1206. In operation, the client interfaces operate according to protocol by which the client servers requests storage transactions by commands which carry parameters sufficient for storage domain identification, including for example an identifier of an initiator, a logical extent such as a LUN number, and an identifier of a target device. The storage server 1200 maps in the requested transaction to a virtual device, which in turn allocates physical storage for use in the transaction from among the physical storage devices. The storage server 1200 also includes resources that emulate the target physical device identified in the request. The storage server 1200 is able to direct storage transactions using local configuration data, and simplify the management of storage for the client servers.

Detailed Description Text (6):
FIG. 1 illustrates a management interface 108 coupled to server 1200 via communication link 109. The communication link, served by interfaces in the station 108 and in the server 1200, comprises for example, an Ethernet network link, a serial cable coupled to serial ports, or an internal bus interface in various embodiments.

Detailed Description Text (7):
Communication between the servers 1201-1203 and the storage devices 1205-1207 is provided via a fibre channel arbitrated loop network through the storage server 1200 as an intermediate device. The channels over FC-AL can be accomplished using a protocol compliant with the standard small computer system interface version 3 (SCSI-3) preferably using a fibre channel medium, also termed fibre channel protocol (FCP) (e.g., SCSI-X3T10 and FCP X3.269-199X). In other embodiments, protocols such as the Internet Protocol are used over the fibre channel fabric carrying storage transactions in a variety of protocols. In some embodiments, the storage server 1200 supports multiple protocols for the data storage transactions.

Detailed Description Text (12):
In FIG. 3, there are a plurality of storage devices 1330 through 1339 illustrated coupled to the storage servers 1300-1302. In the diagram, a variety of symbols are used to represent the storage devices, and to indicate that the network is heterogeneous and can utilize a wide variety of devices managed by the virtual device interfaces at the servers 1301 through 1302. Also, the communication channels can be varied. Thus, hubs 1340, 1341 and 1342 are included in the network to facilitate a variety of communication protocols between the storage devices and the storage servers.

Detailed Description Text (15):
The storage server 102 has connection options 130 including a set of communication interfaces adapted for users and for other data processing functions, and storage options 128 including a set of communication interfaces adapted for storage devices. The storage server 102 has a hardware interface 126, an operating system 124, a block storage interface 118, a management interface 120, and a protocol interface 122. The connection options 130 include serial connections 140, a front panel connection 142 supporting a configuration management routine in one embodiment, an Ethernet connection 144 supporting communication with a remote management station, and a network interface 146. The storage options 128 include the drive array 132, the solid state drive (SSD) 134, the SCSI interface 136, and the network interface 138. The SCSI interface 136 is coupled to a DVD/CD-R 148. The network interface 138 is coupled to a storage server 102G and/or storage 150.

Detailed Description Text (16):
The connection options 130 are various methods of connecting servers and clients to the storage server 102. The serial connections 140 support network management, modems for remote management, and uninterruptible power supply messages. The front panel connection 142 supports a management connection with the front panel display of the storage server 102. The Ethernet

connection 144 supports an Ethernet interface for management protocols and possibly for data transfer. The network interface 146 is one of potentially many high speed interfaces on the server. In some embodiments, the network interface 146 is a fibre channel interface with drivers for a fibre channel arbitrated loop (FC-AL). The network interface 146 may also include drivers for SCSI-3 over the fibre channel medium using fibre channel protocol (FCP).

Detailed Description Text (17):
The hardware interface 126 provides interface specific hardware components. For example, the network interface 146 has a network interface specific set of software modules to support configuration, diagnostics, performance monitoring, and health and status monitoring.

Detailed Description Text (18):
The operating system 124, the tables 116, and the interfaces 118-122 support the virtual device and storage routing functionality of the storage server 102. These components of the storage server 102 route storage transactions among appropriate storage options 128 and the connection options 130 using configured sets of driver modules in the system.

Detailed Description Text (20):
The block storage interface 118 provides software modules to support block data transfers. The interface 118 includes support for striped data storage, mirrored data storage, partitioned data storage, memory cache storage, and RAID storage. The different supported storage types can be linked to form various combinations such as a mirrored data storage with a memory cache.

Detailed Description Text (21):
The protocol interface 122 provides software modules for translating and responding to requests in a variety of protocols. One set of modules is provided for the layers of an Ethernet connection: the hardware driver, the data link driver, the Internet protocol (IP) driver, the transmission control protocol (TCP) driver, the user datagram protocol (UDP) driver, and other drivers. Another set of modules provides drivers for FCP.

Detailed Description Text (22):
The management interface 120 provides software modules for managing the storage server 102. The management interface 120 contains interfaces for managing access to the tables 116. The management interface 120 also contains interfaces for rules-based management of the system including: scheduling, or process orchestration; monitoring the system; informed consent management; and handling system processes and events. The informed consent management module is premised on providing rules based management suggestions for configuring and maintaining the storage server 102.

Detailed Description Text (27):
The storage server 102A will map the SCSI-3 storage transaction request to a virtual circuit corresponding to a virtual LUN. A virtual circuit is a sequence of one or more virtual devices. A virtual device is composed of one or more devices such as software modules or hardware components. For example, two network interface devices could be combined to be a virtual device. Similarly, two cache devices could be combined as a virtual device. This design permits components to fail without disrupting the storage transaction processing capabilities of the storage server 102.

Detailed Description Text (30):
The final virtual device in a virtual circuit is typically the format translation and communication channel drivers for controlling the storage. For example, the drive array 132 is controlled by redundant hardware driver modules (HDMs) that are grouped to form a virtual device. The HDMs provide BSA to SCSI translation and the HDM handles the interface to the drives that compose the drive array 132. Similarly, if the virtual circuit is a link to some other type of storage over the network interface 138, there will be a virtual device with support for BSA translation to the storage device communication channel protocol.

Detailed Description Text (31):
The storage server also includes resources in the operating system and at the interfaces to the client servers which emulate physical storage devices. The emulation allows the virtual devices to appear to the client servers accessing the storage as if they were physical devices. Thus, the client servers can be configured to communicate using standard protocols, such as FCP using SCSI commands for storage transactions. In the embodiment utilizing SCSI commands, the

emulation involves responding to an inquiry command according to the SCSI protocol with device identifiers and device capability information expected by, or compatible with, the initiating server. Also, a read capacity command and a mode page data command in the SCSI protocol are handled by the emulation resources in a manner that allows the client servers using the storage to rely on standard configuration information for physical storage devices, while the storage server spoofs the client server by emulating the physical storage devices at the interface with the client server, and maps actual storage transactions to virtual devices. The emulation resources also allow the virtual devices to be identified by the combination of an initiator, a logical unit number (LUN), and a target device identifier, without requiring the storage transaction to be tied to the specific physical target device identified in the request.

Detailed Description Text (32):
FIG. 5 is a block diagram showing functional components of a server, like that illustrated with respect to FIG. 4, acting as a storage management system 151 for use in storage domain management. The system 151 includes a storage manager operating system 152. With the storage manager operating system 152, functional components include storage domain routing resources 153, legacy device emulation resources 154, data migration resources 155, and redundancy, hot swap and failover resources 156. The storage manager operating system coordinates communication among these resources an on-chassis cache 157, a management interface 158, and in this embodiment an on-chassis storage array 159.

Detailed Description Text (34):
A plurality of communication interfaces 160-165 are provided on the system 151. In this example, the interface 160 is adapted to execute protocol X between a client and the storage management system 151; interface 161 is adapted to execute protocol Y between a client and the storage management system 151; interface 162 is adapted to execute protocol Z between a storage device and the storage management system 151; interface 163 is adapted to execute protocol A between a storage device and the storage management system 151; interface 164 is adapted to execute protocol B between a storage device and storage management system 151; and interface 165 is adapted to execute protocol C between the storage manager system 151 and another storage management system on the network.

Detailed Description Text (36):
Storage transactions traverse the interfaces 160-165 from respective communication media to the internal resources of storage management system 151. In a preferred system, storage transactions are translated to a common messaging format internal to the system for routing among the various interfaces, independently of the protocols executed by those interfaces. Storage domain routing resources 153 map the transactions within the storage domain using virtual circuits configured for particular client devices and storage devices. Legacy emulation resources 154 and data migration resources 155 allow a storage domain to be reconfigured at the storage management system 151 as new equipment is added and removed from the network. For example, a new storage device may be added to the network, and a data set in an existing storage device may be migrated to a new storage device, and storage transactions from clients using the data set may be made to appear as if they remain on the existing storage device during the migration, and after migration is completed by providing target emulation. The redundancy, hot swap, and failover resources 156 insure fault-tolerance, and support continuous operation of the storage management system 151 for high throughput data storage networks.

Detailed Description Text (39):
FIG. 6 includes the storage server 102A. The storage server is designed to provide a high degree of redundancy while using standard components and the standard based devices. For example, the storage server 102A uses a high speed version of the standard peripheral component interconnect (PCI) implementation and standard fibre channel arbitrated loop (FC-AL) interfaces. A variety of other protocols and interfaces can be used in other embodiments.

Detailed Description Text (40):
The storage server 102A has four separate 64-bit 66 MHz PCI busses 200A-D. Many different configurations of storage devices and network interfaces in the slots of the PCI busses are possible. In one embodiment, the PCI busses are divided into two groups: the SSD PCI busses 200A-B and the interface PCI busses 200C-D. Each group has two busses that are designated by the terms upper and lower. The upper and lower busses in each group can be configured to provide redundant services. For example, the lower SSD PCI bus 200B has the same configuration as the upper SSD PCI bus 200A.

Detailed Description Text (43):
The interface PCI busses provide an interconnection from the network interface controller (NIC) modules 206A-B, the redundant arrays of independent disks (RAID) Controller (RAC) modules 212A-B, and application specific processing (ASP) modules 208A-D to the HBC modules 202A-B.

Detailed Description Text (47):
The front panel display (FPD) 220 provides a user interface for the storage server 102A. The FPD contains a display device and an input device. In one embodiment, a touch sensitive liquid crystal display (LCD) is used to present a touch sensitive screen with input capabilities. The FPD 220 is coupled to the HBC modules 202A-B to support status displays, configuration display and management, and other management functions.

Detailed Description Text (50):
The bus system includes all of the busses in the storage server 102A. In this example, the bus system includes the four PCI busses interconnected by the host bridge controllers. The bus system also includes the PCI busses internal to the HBC modules that provide additional interfaces. The slots include all of the positions on the bus system which can receive interfaces. In this example, each of the four PCI busses outside of the HBC modules can accommodate four interfaces.

Detailed Description Text (51):
The interfaces are the cards or other devices that are placed in the slots. The interfaces support drivers and hardware for the data stores coupled to the interfaces.

Detailed Description Text (60):
FIG. 7 includes the following operating system components: the hardware interface module 900, the Nucleus PLUS.TM. real-time kernel module 902 available from Accelerated Technologies, Inc., Mobile, Ala., the ISOS protocol management module 904, and the storage services module 906. The hardware interface module 900 allows the software components of the storage server to communicate with the hardware components of the storage server.

Detailed Description Text (62):
The ISOS module 904 permits the storage server to support a messaging architecture for input and output. The hardware modules such as the RAID controller (RAC) modules, the network interface controller (NIC) modules, the solid state drive (SSD) modules, the disk drive hub (DDH) modules, and the fibre channel hub (FCH) modules, are all input/output processors (IOPs). The master host bridge processor (HBC) module serves as the host.

Detailed Description Text (69):
FIG. 8 illustrates a network interface card 500 having an HDM 504. The card 500 has a physical interface 501 to a fibre channel network. A network interface chip 502, in this example a Qlogic device, such as an ISP 2200A provided by Qlogic Corporation of Costa Mesa, Calif., is coupled to the physical interface 501. The network interface chip 502 generates communication represented by line 503, which is processed in the HDM 504. The HDM 504 conditions the communications for use by other driver modules in the system. Thus, communication represented by line 505 has an SCSI format. Communication represented by line 506 has a message format such as a BSA format. Communication represented by line 507 has an Internet Protocol (IP) format. The HDM is an instance of a driver class labeled "QLogic driver" in the diagram, and given device identifier DID 401 in this example. The physical interface is identified as NIC #1.

Detailed Description Text (70):
FIG. 9 illustrates a storage device 700 which is implemented by an array of non-volatile integrated circuit memory devices. One embodiment of the device 700 is described in co-pending U.S. patent application Ser. No. 09/292,536, entitled High Speed Bus Interface for Non-Volatile Integrated Circuit Memory Supporting Continuous Transfer, filed Apr. 15, 1999, which is owned by the same assignee as the present application, and is incorporated by reference as if fully set forth herein. The HDM 702 is coupled with the array 701, and translates the block storage architecture communications on line 703 into format for storage and retrieval from the array 701. In this example, the HDM 702 is given a device identifier 1130. The physical interface is identified as SSD #4.

Detailed Description Text (72):

HDMs 801 and 802 are connected with physical fibre channel arbitrated loop connections 803 and 804, respectively. The HDM 801 is given device identifier 1612 and the HDM 802 is given device identifier 1613. The connection 803 is coupled to a fibre channel interface 805. Interface 805 includes a network interface chip 806 which is coupled with physical interface 820, and to an HDM 807. An ISM 808 is coupled to the HDM 807 and to the internal communication path 809. The ISM 808 translates the block storage architecture communications on line 809 into IOCB communications for the HDM 807. The HDM 807 communicates with the network interface chip 806, which in turn drives the fibre channel 803. The ISM 808 is given device identifier 1210, and the HDM 807 is given device identifier 1110. The physical interface 805 is labeled RAC #0.

Detailed Description Text (73):
The fibre channel connection 804 is coupled to interface 810. Interface 810 has a configuration like interface 805. Thus the interface 810 includes a physical fibre channel interface 811 which is driven by network interface chip 812. The network interface chip 812 communicates on the channel represented by line 813 with HDM 814. HDM 814 communicates with ISM 815 via channel 816. The ISM 815 manages an interface to the BSA format messages on channel 817. In this example, the ISM 815 is given device identifier 1211. The HDM 814 is given device identifier 1111. The interface 810 is identified as RAC #1.

Detailed Description Text (74):
FIGS. 8-10 illustrate a variety of physical communication interfaces and corresponding HDMs. FIGS. 11-14 illustrate a variety of ISM examples according to the present invention, which can be configured into data paths.

Detailed Description Text (75):
FIG. 11 shows an SCSI target server 550, which is one example of a protocol server module according to the present invention. Similar protocol server modules can be implemented for any particular storage channel or network protocol implemented by users of the data managed through the storage server of the present invention. The target server 550 has a message interface 551 which receives incoming messages from an HDM, such as the HDM of FIG. 8, coupled to a communication interface adapted for connection with a user. In this example, the messages on interface 551 have an SCSI format. In other examples, the messages may already have the BSA architecture, or some other architecture which is suitable for the protocol on the communication interface being served. The server 550 includes a switch function 550 which translates incoming messages to an SCSI to BSA translator 553, or to an answer local function 554. Typically, messages are forwarded by the translator 553 as outgoing messages on line 555. Incoming messages on line 555 are supplied to translator 556 which translates the incoming BSA messages to the SCSI format used on line 551.

Detailed Description Text (77):
The target server 550 in this example is an instance of a class SCSI target server, and given a device identifier 500. One function of the protocol server, such as the SCSI target server 550, is to identify the storage extent which is subject of a storage transaction on the associated interface. The storage extent is mapped to a virtual circuit using the configurable logic in the storage server as described in more detail below.

Detailed Description Text (78):
FIG. 12 illustrates an ISM 650 which performs a mirror management data path task. The ISM 650 includes an interface 651 which is connected to the internal communication channels on the device. Logic processes 652 receive the incoming communications and data and manage a mirroring function. The logic 652 communicates with a plurality of drive interfaces including primary drive 653, secondary drive 654, tertiary drive 655, and standby drive 656. Although 3-way mirroring is shown in the diagram, any number of mirror paths may be implemented for "n-way" mirrors using virtual circuits. Although the drive interfaces in FIG. 12 are labeled with the term "drive," other types of storage devices can be used in the mirroring functions. The drive interfaces 653-656 communicate using the internal communication channels with the HDM modules associated with the target storage devices used in the mirroring function, or with other ISM modules as suits the particular virtual circuit. In this example, the mirror ISM 650 is implemented as an instance of a class "mirror," and given a device identifier 10200.

Detailed Description Text (79):
FIG. 13 illustrates a partition ISM 750. The partition ISM 750 includes an interface 751 which receives internal communications from other driver modules, and an interface 752 which also

communicates with other driver modules. The ISM 750 includes logic processes 753, data structures for storing a base address 754 and a limit address 755, and a drive interface 756. The partition logic process 753 configures the subject storage device identified by the drive process 756, using a logical partitioning function useful for a variety of storage management techniques, so that the physical device appears as more than one logical device in the virtual circuits. In this example, the partition ISM 750 is an instance of a class "partition," and given device identifier 10400.

Detailed Description Text (80):
FIG. 14 illustrates a cache ISM 850. The cache ISM 850 includes logic processes 853 which communicate with an interface 851 to the internal message passing structure on the storage server. Data structures in the cache ISM 850 include a local cache memory allocation 854, a cache table 855 which identifies the data stored in the cache 854, and a drive interface 856. The drive interface communicates on channel 857 with an HDM associated with the particular virtual circuit being served by the cache. The cache memory 854 in one embodiment is managed locally in the storage server. In an alternative embodiment, the cache can be stored in a high-speed, non-volatile memory, such as a solid state memory module having architecture Like that described with respect to FIG. 9. In the preferred embodiment, the cache ISM 850 is implemented as an instance of a class "cache," and given a device identifier 10300.

Detailed Description Text (81):
FIG. 15 provides a heuristic diagram of redundant virtual circuits implemented by data paths including a plurality of driver modules according to the present invention. Virtual circuits include an external interface for communication with a user of the data, a protocol translator for translating communications with the user into the communication format of the driver modules, and a storage object which includes a communication interface to a storage device. Storage operators which perform data path tasks can exist between the translator and the storage object. The optimal ordering of the driver modules acting as storage operators, such as cache, mirror, partition, etc., is done by the system designer using the configurable logic provided by the storage server.

Detailed Description Text (82):
In the example illustrated in FIG. 15, the external interface is provided by the NIC #0, and its associated HDM is represented by block 1010. The protocol translator is provided by the SCSI target server ISM 1011. A cache function is provided by the ISM 1012. A mirror function is provided by the ISM 1013. The storage objects are accessed from the mirror function 1013, and consist of a set of physical storage interfaces selected in this example from the fibre channel basic daisy chain interface and its associated HDM represented by block 1014 or an external LUN interface, the disk drives in the fibre channel arbitrated loop accessed through the ISM/HDM pair represented by block 1015 and the redundant block 1016, the solid state storage device and its associated HDM represented by block 1017, and the interface to an external disk drive and its associated ISM/HDM pair represented by block 1018. Separate HDM modules on the fibre channels interface to disks (01), (02), (03), and (04) manage the communication across the fibre channel arbitrated loops with the interfaces 1015 and 1016.

Detailed Description Text (83):
In the embodiment shown, the mirror module 1013 accesses disks (01), (02), and (04) as the primary, secondary and standby drives, respectively, for the mirror functions. Although the mirror module shown in FIG. 12 includes the tertiary drive interface, this tertiary drive is not used in the example system.

Detailed Description Text (85):
A redundant data path is implemented using the interface NIC #1 and its associated HDM represented by block 1025, the SCSI target server ISM represented by block 1026, the cache ISM represented by block 1027, and the mirror ISM represented by block 1028. Redundancy in the data storage devices is accomplished using the mirror function. The redundant driver modules are distributed in a preferred embodiment on separate IOPs within the storage server.

Detailed Description Text (87):
In the preferred system, the configuration tables are managed by a persistent table driver, such as that illustrated in FIGS. 16 and 17. Referring back to FIG. 4, the storage server 102 stores management and routing information in tables such as the tables 116. The tables 116 can be accessed through the management interface 120. The tables 116 will typically be stored in

persistent memory such as a non-volatile memory. The tables 116 can be maintained redundantly to provide failsafe support.

Detailed Description Text (90):
The persistent data storage maintains a wide variety configuration information for the system. The DDM roll call table 1409 includes a list of all the instances of the device driver modules, and their unique device IDs. The storage roll call table 1411 includes a list of all the active storage devices detected by the storage server. The roll call tables can be utilized by the virtual device table 1410 and by the configuration tools to create virtual circuits. The LUN export table 1407 provides a technique for mapping the identified storage extents within a storage channel transaction to virtual circuits. The external LUN table 1413 identifies logical units of storage which are maintained in other storage servers connected through the external storage interface on the storage server.

Detailed Description Text (93):
The export table 1407 maps addressing information received with a storage transaction to a virtual circuit or to a storage option. In the case of SCSI-3 over a fibre channel interface, the addressing information used is the initiator ID, the target LUN, and the target address.

Detailed Description Text (106):
The connection option such as the network interface 146 over which the storage transaction is received is coupled to a hardware device driver. The hardware device driver receives the storage transaction and depending on the protocol, dispatches it to an appropriate virtual device for handling that protocol.

Detailed Description Text (108):
The SCSI target device driver further analyzes the request. The first part of the analysis is to determine which virtual circuit to map the request to. This determination can be made using the information in the export table. In this example, Table 1, indicates that a request using the SCSI protocol specifying LUN 2 should be routed to the virtual circuit starting with the virtual device 12. In one embodiment, all of the SCSI target requests are routed to the same SCSI target driver for a single interface. In this embodiment, the parameter information for the target VD 12 is used to control the behavior of the SCSI target device rather than routing the message to a second virtual device for a SCSI target.

Detailed Description Text (113):
Accordingly, within the storage platform, hardware functions (such as disk or flash storage) and software functions (such as RAID stripes or Mirrors) are all accessed via software drivers most commonly referred to as devices.

Detailed Description Text (124):
User Interface

Detailed Description Text (128):
The user launches a host manager using button 1405. This section describes a Java based user interface (UI) for defining hosts (servers) to storage server. The management software opens a window, such as the window 1450 shown in FIG. 19, that presents a table 1451 with entries that contain a host name in column 1452, a port number in column 1453, an initiator ID in column 1454 and a description in column 1455 for each host available for configuration and use. Other fields include a network interface card identifier in column 1456, and a unique host identifier in column 1457. The unique host identifier in a preferred example is the World Wide Number value for a fibre channel host.

Detailed Description Text (130):
FIG. 20 illustrates a dialog box 1465 used in response to the add new host button 1458 of FIG. 19. The dialog box 1465 includes a field 1466 for inserting a host name, and a field 1467 for inserting a unique identifier of the host. Also fields are provided for inserting information about the network interface card 1458, a port number 1459, and an initiator ID 1470. Finally, a description field 1471 is included.

Detailed Description Text (131):
FIG. 21 illustrates a dialog box 1475 used for modifying a host. It includes the fields 1476 for changing a host name, 1477 for changing a host unique identifier, 1478 for changing the

network interface card identifier, 1479 for changing a port number, and 1484 for changing an initiator ID associated with the host. Also, a field 1481 is included for inserting or changing description text.

Detailed Description Text (133):
The User Interface consists of menus and a table, or other graphic construct, for displaying the host information. When the user enters the host manager panel, the table is populated with all the existing hosts. The user may select a row in the table. Each row contains information about one host. The user may then choose to modify or delete the host. If modify is chosen, a dialog box will appear allowing the user to change the host name and/or the description. The user will then hit the OK or Cancel button. If OK is hit, the changes will appear in the table and be sent to the server. If delete is chosen, a dialog box will appear with a label indicating the host to be deleted and buttons for OK and Cancel. If OK is hit, the host row will be deleted from the table and the deletion will be made at the server. If Add is chosen, a dialog box will appear that enables the user to add all information about a host. If OK is hit, a new row will be added to the table for that new host and a add will be done at the server. Clicking the column label will sort the columns.

Detailed Description Text (146):
Add and Delete functions are provided to create or remove entries, as well as a Modify function so that using tools provided by the user interface, the user can change things like "owner" or "last serviced" or "description", etc. fields for storage elements in the tree. The user will specify what it is that they are adding (mirror, stripe, disk, etc.), and an appropriate set of controls will be given them.

Detailed Description Text (164):
In Stage 1, the storage server 10 maps all data access requests identifying the data set subject of the transfer and received on interface to link 13, to the link 14 for connection to the device 11, which stores the data set subject of the request. The storage server receives a control signal initiating a hot copy process and identifying a target device, in this example the device 12. This step initiates Stage 2, during which the data set is transferred as a background process from the first device 11 through the storage server 10, into the second device 12. Parameters are maintained on the storage server 10 indicating the progress of the transfer of the data set, and indicating a relative priority of the background hot copy process with respect to the data access requests from the client processor. During the hot copy process, data access requests are mapped to the first device 11 and the second device 12 depending on the progress of the hot copy, and on the type of request. Also, the storage server includes resources for assigning a priority to the hot copy process. If the priority of the hot copy process is low, then the client processor does not experience significant delay in fulfillment of its data access requests. If the priority of the hot copy processes relatively high, then the client processor may experience some delay in fulfillment of its data access requests, but the hot copy process will complete more quickly.

Detailed Description Text (168):
FIGS. 28, 29, 30 and 31 illustrate various aspects of a software implementation of a hot copy process for execution in the intelligent network server described above. In other storage servers used for a hot copy process, variations in the implementation will be made to accommodate the particular system. More details of components of a virtual circuit, persistent table storage, and user interface structures are described with reference to FIG. 9, and the following figures.

Detailed Description Text (186):
FIG. 32 illustrates a virtual circuit according to the present invention implemented using an input/output processor 2000 that combines a network interface (NIC) and a RAID controller interface (RAC) on single card. In this embodiment, a target server 2001 includes or is coupled with local answer routines according to the present invention.

Detailed Description Text (187):
In operation, the target server 2001 emulates the target 2027 to preserve the configuration of the host 2010 as much as possible. Thus the configuration record 2012, or other record in the system coupled to the interface to host 2010, includes a flag indicating that the target server 2001 has been mapped to a legacy device. In this case, incoming storage channel commands are monitored. All commands which can be answered locally are handled using locally stored data

about a legacy device. Read and write commands are passed through to the legacy device 2027, or to such other device as is configured as part of the virtual circuit assigned to the logical address of the legacy device.

Detailed Description Text (208):
The storage domain manager offers a comprehensive set of centralized storage management capabilities that can be leveraged from a single management interface, across all attached servers and storage, regardless of vendor. From a central location, a system administrator may control the movement or mirroring of data between heterogeneous storage resources, and can dynamically leverage these capabilities across different heterogeneous storage resources over time. This results in a significant cost savings and simplification of administrative complexity. As a scalable, intelligent platform, the storage domain manager resides in the perfect central location to host storage management functionality that can be leveraged across all attached server and storage resources.

Current US Cross Reference Classification (1):
707/1

Current US Cross Reference Classification (2):
707/10

Current US Cross Reference Classification (3):
707/102

CLAIMS:

3. The method of claim 1, wherein said assigning storage resources includes configuring said storage resources in the network to the logical storage extents using a graphical user interface at the intermediate system.

8. The method of claim 6, wherein said defining a storage domain includes configuring said storage resources in the network to the logical storage extents using a graphical user interface at an intermediate position between the clients and the storage resources.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

□ Generate Collection Print

L14: Entry 3 of 7 File: USPT Jul 29, 2003

DOCUMENT-IDENTIFIER: US 6601072 B1
TITLE: Method and system for distribution of application data to distributed databases of dissimilar formats

Brief Summary Text (4):
In a relational database, data are stored in multi-dimensional tables. Each table has multiple rows and columns populated with information that describe the relationship between the data elements. Use of a relational database application, instead of a proprietary database, has the obvious advantage of opening the data to the vast array of relational database query and data manipulation tools that are pervasive among enterprise computing developers. It also allows the business user and the application itself to take advantage of the proven reliability of a commercial enterprise database engine. A relational database is not tied to a specific record and file format; customers' writing tools that interface with the database will be significantly less impacted when changes and additions are made to the database by the application developers.

Brief Summary Text (10):
Business objects represent a software application that needs to view or manipulate data stored in the software application's database. Business objects create data objects via a factory object. Factory objects are abstract factories that produce data objects of different types. Data objects represent some logical grouping of information that is applicable to an application. For instance, a data object representing a person for an address book application could contain all relevant information about that person that the address book tracks. Storage objects present the interface to software applications or business objects. Data access objects, or converters, have the responsibility of converting data of a data object into a different format that can be used to store the data object's encapsulated information into a database. Database interface objects receive data from the data access objects and store that data into physical databases.

Brief Summary Text (11):
Business objects have methods defined to interface software applications. When a software application is ready to write data into a database, the business object requests creation of a data object from a factory object. The data object is created and the business object then populates the data object with information from the software application. After the data object is filled with information, the business object requests the storage object to store the information in this data object into a plurality of external databases and passes this data object to the storage object. The storage object notifies local and remote data access objects about the data, and these data access objects then store the data object into their respective databases through a database interface. The data can be stored in different formats into different databases.

Detailed Description Text (3):
A computer software database application runs on the computer 3 and accesses the data stored in the local database 8; similar database software residing on the server 10 accesses data stored in the remote database 9. A local software application program can access the remote database across the network 11. The local database 8 can have a different format than that of the remote database 9. For this description, the local database 8 is considered to be a proprietary database, while the remote database 9 is considered to be a relational or object-oriented database. The computer software access application is generally developed using an object-oriented design approach, and it uses objects to represent different computing elements and to encapsulate their interface functions.

Detailed Description Text (12):

The storage object 24 provides the interface to a storage subsystem. The storage subsystem is responsible for distributing the data to converters that reform the data into an appropriate format for storage in a database of a given type. The storage object 24 is also responsible for maintaining a list of subscribing data access objects and publishing database update requests to each of them when directed to do so by a business object 12. The business object 12 interacts with the storage object 24 when it needs to access a database. The storage object 24 has the following exemplary methods defined: attach( ), detach( ), notify( ), write( ) and read ( ). The storage object 24 is also a singleton object.

Detailed Description Text (14):

Local data access object (DAO) 34, and remote DOA 36, are "observers" in this description. Local DAO 34 generally resides on the same platform as the storage object 24. Remote DAO 36 can reside on a platform that is separated geographically from the storage object 24. This remote platform can be running on an operating system that is different from the operating system on the local platform. Remote DAO 36 can exist as stand-alone server process and have additional logic to handle requests made across the network 11. DAOs 34, 36 have the responsibility for converting data between a data object form and a format that is used to store the data object's encapsulated information into a database. DAO 36 has a published remote interface called a stub that is available to the storage object 24 and that allows the data object 20 to be serialized and passed to a remote DAO 36. When a remote DAO 36 is instantiated, part of its configuration includes the address of the machine running the main application containing the business object 12. It registers with its storage object 24 when it is ready to receive update requests, and creates whatever additional objects are necessary for it to communicate with its target database through its interface with the database. In this description both the local DAO 34 and the remote DAO 36 are defined with the following methods: readAttributes( ), storeAttributes( ) and format( ).

Detailed Description Text (15):

The DAOs 34, 36 "flatten" the data object 20 by converting the information into an appropriate format before passing it to keyed file object 38 or other proprietary database interface if the data is to be stored in a proprietary database file, or to a database engine object 40 if the data is to be store in a relational or object-oriented database. Proprietary database interface object 38 is an object that owns and manages low level input/output (I/O) for a specific proprietary database file. Proprietary database interface object 38 usually has read( ) and write( ) methods defined to handle the data file process commands presented by DAO 34. Database engine object 40 is an object representing a real database engine application that processes SQL (Structured Query Language) commands presented by DAO 36. SQL is a standard interactive programming language for accessing information from and updating a database. Queries take the form of a command language that enables selecting, inserting, updating and locating data. Database engine object 40 has select( ), inserts and updates methods to handle a relational database.

Detailed Description Text (17):

The following sequence of steps, performed in response to a write data request, correspond to the step number shown on the left-hand side of FIG. 3. Step 1: Business object 12 requests a data object's attributes to be written to the databases by calling StorageObject::write(DO) and passing the data object (DO) 20. Step 2: StorageObject::write(DO) calls StorageObject::notify (DO) again passing the DO 20. The job of the Notify method is to publish the updated DO 20 to all subscribed data access objects 34, 36. Step 3: Since, in the present case, the local version of the database is the primary version, its database is updated first. Notify(DO) does this by calling o->storeAttributes(DO) where "o" is a pointer to the local DAO object 34. Step 4: In this example, the local DAO 34 interfaces with a keyed file object 38 which manipulates a specific proprietary database file. LocalDAO::storeAttributes(DO) calls LocaIDAO::format(DO) to flatten the DO's attributes into a record which contains a key and is stored in a buffer. The buffer is returned to LocalDAO::storeAttributes(DO). Step 5: LocalDAO::storeAttributes(DO) then calls LocalKeyedFileObject::write(buffer) and passes the formatted buffer. The write method reads the key from the first "n" bytes of the record and extrapolates a physical record to replace in the keyed file. The write buffer then writes the record to that location. If the record is locked by another process or some other error occurs, then an error object is created and returned to the local DAO 34. Once the operation is complete, control returns to StorageObject::notify(DO). Step 6: StorageObject::notify(DO) now invokes the remote DAO's 36 storeAttributes method through its published remote stub interface and again passes the DO 20. The stub serializes the passed data object and transmits the serialized data stream to the remote DAO, where the data stream is deserialized back into an instantiated data object (DO). Step 7: RemoteDAO::storeAttributes(DO) calls its format method to flatten the DO's attributes into one or more SQL update commands which are returned to storeAttributes in an array of buffers. Step 8: storeAttributes(DO) then iterates through the array of buffers and invokes each SQL update command such that the database engine 40 can process the request and update the database.

Detailed Description Text (21):
Although not critical to the present invention, storage objects 24 are preferably created by factories and have virtual base classes that define their common behavior. Data access objects 34, 36 are also preferably created by factories and multiply inherit from two virtual base classes. The first base class defines behavior common to local DAOs 34 or remote DAOs 36. The second base class defines behavior common to the database type with which the DAO 34, 36 interface.

Detailed Description Text (23):
FIG. 4 illustrates an exemplary diagram of the relationship between physical databases, derived data access objects and abstract base classes of the storage subsystem. The numbers shown parenthetically in this description refer to reference numerals depicted in FIG. 4. This highly simplified diagram shows only two types of data objects 21, 23 and two types of databases 37, 39 (associated with physical relational database 43 and physical proprietary database 41, through their respective database interfaces). Also depicted are four data access objects - DAO1 (29), DAO2.(31), DAO3 (33), and DAO4 (35).. The solid lines with triangular arrowheads between the DAOs and the database types and data object types indicate inheritance. Thus, in the situation depicted, DAO1 (29) inherits from RelationalDatabase (37) and DataObjectType1 (21). DAO2 (31) inherits from ProprietaryDatabase 39 and from DataObjectType1 (21). DataAccessObject 3 (33) inherits from RelationalDatabase 37 and from DataObjectType2 (23). Finally, DAO4 (35) inherits from PropietaryDatabase 39 and from DataObjectType2 (23). As further illustrated in FIG. 4, DAO1 (29) and DAO3 (33) are associated with physical relational database 43 through a database interface (not shown). Likewise, DAO2 (31) and DAO4 (35) are associated with physical proprietary database 41 through a proprietary database interface (not shown).

Detailed Description Text (25):
A Jini system consists of services that can be collected together for the performance of a particular task. Services may make use of other services, and a client of one service may itself be a service with clients of its own. Services in a Jini system communicate with each other by using a service protocol, which is a set of interfaces written in the Java programming language. The set of such protocols is open-ended. Communication between services can be accomplished using the Java Remote Method Invocation (RMI). The infrastructure to support communication between services is not itself a service that is discovered and used; rather it is a part of the Jini technology infrastructure. RMI provides mechanisms to find, activate, and garbage-collect object groups. Fundamentally, RMI is a Java-programming-language-enabled extension to traditional remote procedure call mechanisms. RMI allows not only data to be passed from object to object around a network, but full objects, including code. Much of the simplicity of the Jini system is enabled by this ability to move code around a network in a form that is encapsulated as an object.

Detailed Description Text (26):
In view of the foregoing discussion, there are two possible implementation strategies for enabling the remote DAOs: 1. A small DAO interface object can reside in the local subsystem that acts as a proxy for the remote DAO. The local proxy DAO can setup a TCP/IP socket connection to the remote DAO and then serialize the data object's attributes and pass them over the connection. 2. The remote DAOs could be implemented as Java applets and a thin JNI (Java Native to Interface) style interface layer could be written under the main application to convert the C++ data objects into Java Beans, or other Java-based object types. Data objects passed from a storage object to the remote DAO's local stub are then serialized and sent over the network to the remote DAO applet with a remote method call.

Detailed Description Text (28):
With respect to the second implementation strategy, a disadvantage to this approach is that the application may need to be written in two different languages. However, the Java Native Interface (JNI) was created specifically for the purpose of integrating Java into C++

applications.

Current US Original Classification (1):
707/103R

Current US Cross Reference Classification (1):
707/100

Current US Cross Reference Classification (2):
707/101

Current US Cross Reference Classification (3):
707/4

CLAIMS:

1. A method for distributing application program data to a plurality of databases of dissimilar formats, said method comprising: defining a business object to represent a software application that needs to write data to one of said plurality of databases; defining a plurality of data access objects, each capable of writing data to one of said plurality of databases in its respective format; defining a storage object to represent an interface between said business object and said plurality of data access objects; in response to a write request to one of said plurality of databases from said business object, generating a data object for data to be written to one of said plurality of databases; sending said data object to each of said plurality of data access objects by said storage object; converting data encapsulated in said data object by said plurality of data access objects into a respective format appropriate for each of said plurality of databases; and storing said encapsulated data into each of said plurality of databases according to its respective format by said corresponding data access objects.

10. A computer readable medium containing a program product for distributing program data to a plurality of databases of dissimilar formats, said program product comprising: program instructions for defining a business object to represent a software application that needs to write data to one of said plurality of databases; program instructions for defining a plurality of data access objects, each is capable of writing data to one of said plurality of databases in its respective format; program instructions for defining a storage object to represent an interface between said business object and said plurality of data access objects; in response to a write request to one of said plurality of databases from said business object, program instructions for generating a data object for data to be written to one of said plurality of databases; program instructions for sending said data object to each of said plurality of data access objects by said storage object; program instructions for converting data encapsulated in said data object by said plurality of data access objects into a respective format appropriate for each of said plurality of databases; and program instructions for storing said encapsulated data into each of said plurality of databases according to its respective format by said corresponding data access objects.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

□ Generate Collection Print

L14: Entry 4 of 7 File: USPT Jan 7, 2003

DOCUMENT-IDENTIFIER: US 6504915 B1
TITLE: Multiple node messaging system wherein nodes have shared access to message stores of other nodes

Brief Summary Text (4):
Messaging systems that provide voice, fax and/or e-mail messaging capabilities are well known. FIG. 1 illustrates an exemplary prior art messaging system developed by Unisys Corporation, the assignee of the present invention. The system of FIG. 1 comprises multiple servers (e.g., an A-series or Clearpath.TM. computer offered by Unisys Corp.) each supporting a network applications platform (NAP), which provides an underlying platform for storage and retrieval of messages, and a messaging application running on the platform. A voice mail application, such as the Unisys Universal Voice Messaging System (UVMS), is an example of a messaging application that runs on the messaging platform. The UVMS application determines how calls to the messaging system are handled, what prompts are played to callers, and which features are available. Such applications typically maintain a database of subscribers who have "mailboxes" on the system. The messaging platform interfaces to a telephone network through a Network Interface Unit (NIU). Received messages are stored by the messaging platform in a local message store, or voice file.

Brief Summary Text (5):
Network Interface Units are available from a number of different vendors. For example, NIUs suitable for use with the present invention include: (a) the Telephony Services Platform available from Unisys Corporation, Blue Bell, Pa.; (b) the Summa Four VCO 80 available from Summa Four, Inc., Manchester, N.H.; and (c) the Voice Frame 2020 available from Harris Corporation, Melbourne, Fla. The Telephony Services Platform (TSP) is most preferred. The difference between it and the other products just mentioned is that the TSP includes a board that plays back digitized voice from a voice file whereas the other products require a separate playback module.

Detailed Description Text (2):
FIG. 3A depicts a first preferred embodiment of a distributed messaging system in accordance with the present invention. As shown, this embodiment includes first and second nodes coupled via a PSTN to a plurality of subscribers, which could include computer-based subscribers 20, telephone-based subscribers 22 and fax-based subscribers 24. Node 1 includes a messaging platform 10a having read/write access, via a first bus or cable 11a, to a disk-based voice file 12'a. Similarly, Node 2 includes a messaging platform 10b having read/write access via a second bus or cable 11b to a second voice file 12'b. In addition, each messaging platform has read-only access to the voice file on the other node. Thus, messaging platform 10a has read-only access, via bus or cable 13a, to voice file 12'b; and messaging platform 10b has read-only access via bus or cable 13b to voice file 12'a. Each messaging platform is also coupled via a bus 16 (e.g., a Small Computer System Interface (SCSI) bus) to at least one, and preferably plural, NIUs, which serve as interfaces to the PSTN. It should be noted that the present invention is by no means limited to the manner in which the messaging platforms 10a and 10b are physically connected to the voice files 12'a and 12'b. That is, any type of bus architecture or cable, or even wireless interface, could be used in carrying out the invention. Moreover, although FIG. 3A shows a single bus 16, multiple buses could be used to connect the NIUs to the NAP system, i.e., each NIU could have a dedicated bus to connect it to the messaging platforms 10a and 10b. Such an architecture eliminates a single point of failure that would otherwise be present.

Detailed Description Text (3):
Each NAP (Network Applications Platform) employed to provide a messaging platform (10a, 10b) comprises a combination of software and hardware that facilitates the development and

deployment of network applications. The network applications provide a service to a subscriber or are used to control network resources. Each NIU (e.g., Telephony Services Platform), on the other hand, comprises a connection between the system and various network lines and trunks. It provides an interface to a Signaling System Number 7 (SS7) network and supports T1 and E1 lines.

Current US Cross Reference Classification (3):
707/1

Current US Cross Reference Classification (4):
707/10

CLAIMS:

20. A distributed voice messaging system, comprising: (a) a first node including a first voice messaging platform and a first voice file, wherein said fist voice messaging platform is provided with read/write access to said first voice file; and (b) a second node including a second voice meshing platform and a second voice file, wherein said second voice messaging platform is provided with read/write access to said second voice file; wherein said first voice messaging platform is further provided with read only access to said second voice file, and said second voice messaging platform is further provided with read only access to said fist voice file; wherein each of said first and second messaging platforms is operatively coupled to a network interface unit (NIU); wherein the first and second nodes' read/write access to the first and second voice files, respectively, is dynamically transferable between nodes such that only one node has read/write access to a particular voice file at any particular time, and wherein, during transfer of a node's read/write access to a voice file, there is no interruption of access to other voice files not involved in the transfer.

Previous Doc Next Doc Go to Doc#

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 21 of 21 returned.

□ 1. Document ID: US 20050125621 A1

L25: Entry 1 of 21 File: PGPB Jun 9, 2005

PGPUB-DOCUMENT-NUMBER: 20050125621
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050125621 A1

TITLE: Systems and methods for the implementation of a synchronization schemas for units of information manageable by a hardware/software interface system

PUBLICATION-DATE: June 9, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Shah, Ashish	Sammamish	WA	US	

US-CL-CURRENT: 711/173

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 2. Document ID: US 20050091667 A1

L25: Entry 2 of 21 File: PGPB Apr 28, 2005

PGPUB-DOCUMENT-NUMBER: 20050091667
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050091667 A1

TITLE: System and a method for presenting items to a user with a contextual presentation

PUBLICATION-DATE: April 28, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
McKee, Timothy P.	Seattle	WA	US	
De Vorchik, David George	Seattle	WA	US	
Sheldon, David Joel	Seattle	WA	US	
Guzak, Chris J.	Kirkland	WA	US	
Moore, Jason Fergus	Redmond	WA	US	
Karatal, Kerem B.	Bellevue	WA	US	
Sierra, Giampiero	Seattle	WA	US	
Peterson, Leonard J.	Woodinville	WA	US	

US-CL-CURRENT: 719/328; 715/764

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 3. Document ID: US 20050091255 A1

L25: Entry 3 of 21 File: PGPB Apr 28, 2005

PGPUB-DOCUMENT-NUMBER: 20050091255
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050091255 A1

TITLE: System and method for storing and retrieving a field of a user defined type outside of a database store in which the type is defined

PUBLICATION-DATE: April 28, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Rajan, Rajeev B.	Kirkland	WA	US	
Raman, Balan Sethu	Redmond	WA	US	
Yan, Kangrong	Issaquah	WA	US	

US-CL-CURRENT: 707/102

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 4. Document ID: US 20050091225 A1

L25: Entry 4 of 21 File: PGPB Apr 28, 2005

PGPUB-DOCUMENT-NUMBER: 20050091225
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050091225 A1

TITLE: System and a method for presenting related items to a user

PUBLICATION-DATE: April 28, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
McKee, Timothy P.	Seattle	WA	US	
De Vorchik, David George	Seattle	WA	US	
Sheldon, David Joel	Seattle	WA	US	
Guzak, Chris J.	Kirkland	WA	US	
Moore, Jason Fergus	Redmond	WA	US	
Karatal, Kerem B.	Bellevue	WA	US	
Sierra, Giampiero	Seattle	WA	US	
Peterson, Leonard J.	Woodinville	WA	US	

US-CL-CURRENT: 707/100

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 5. Document ID: US 20050091183 A1

```
L25: Entry 5 of 21                          File: PGPB                      Apr 28, 2005
```

PGPUB-DOCUMENT-NUMBER: 20050091183
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050091183 A1

TITLE: Type path indexing

PUBLICATION-DATE: April 28, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Cunningham, Conor J.	Redmond	WA	US	
Venkatesh, Ramachandran	Bellevue	WA	US	
Hanson, Eric N.	Bellevue	WA	US	

US-CL-CURRENT: 707/1

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 6. Document ID: US 20050091181 A1

```
L25: Entry 6 of 21                          File: PGPB                      Apr 28, 2005
```

PGPUB-DOCUMENT-NUMBER: 20050091181
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050091181 A1

TITLE: System and method for the presentation of items stored on a computer

PUBLICATION-DATE: April 28, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
McKee, Timothy P.	Seattle	WA	US	
De Vorchik, David George	Seattle	WA	US	
Sheldon, David Joel	Seattle	WA	US	
Guzak, Chris J.	Kirkland	WA	US	
Moore, Jason Fergus	Redmond	WA	US	
Karatal, Kerem B.	Bellevue	WA	US	
Sierra, Giampiero	Seattle	WA	US	
Peterson, Leonard J.	Woodinville	WA	US	

US-CL-CURRENT: 707/1

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 7. Document ID: US 20050063083 A1

```
L25: Entry 7 of 21                          File: PGPB                      Mar 24, 2005
```

PGPUB-DOCUMENT-NUMBER: 20050063083
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050063083 A1

TITLE: Systems and methods for the implementation of a digital images schema for organizing units of information manageable by a hardware/software interface system

PUBLICATION-DATE: March 24, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Dart, Scott E.	Lynnwood	WA	US	
Gibson, Bradley P.	Seattle	WA	US	
Evans, Christopher A.	Sammamish	WA	US	
Hellyar, Paul S.	Kirkland	WA	US	
Vaschillo, Alexander	Redmond	WA	US	
Platt, John C.	Redmond	WA	US	
Glenner, Steve C.	Bellevue	WA	US	
Ballou, Nathaniel H.	Kirkland	WA	US	

US-CL-CURRENT: 360/1

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 8. Document ID: US 20050055380 A1

L25: Entry 8 of 21 File: PGPB Mar 10, 2005

PGPUB-DOCUMENT-NUMBER: 20050055380
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050055380 A1

TITLE: Systems and methods for separating units of information manageable by a hardware/software interface system from their physical organization

PUBLICATION-DATE: March 10, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Thompson, J. Patrick	Seattle	WA	US	
Cameron, Kim	Bellevue	WA	US	
Acharya, Srinivasmurthy P.	Sammamish	WA	US	
Raman, Balan Sethu	Redmond	WA	US	

US-CL-CURRENT: 707/200

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 9. Document ID: US 20050055354 A1

L25: Entry 9 of 21 File: PGPB Mar 10, 2005

PGPUB-DOCUMENT-NUMBER: 20050055354
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050055354 A1

TITLE: Systems and methods for representing units of information manageable by a hardware/software interface system but independent of physical representation

PUBLICATION-DATE: March 10, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Thompson, J. Patrick	Seattle	WA	US	
Cameron, Kim	Bellevue	WA	US	
Smith, Walter R.	Seattle	WA	US	
Shakib, Darren A.	North Bend	WA	US	
Ballou, Nathaniel H.	Kirkland	WA	US	
Celis, Pedro	Redmond	WA	US	

US-CL-CURRENT: 707/100

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 10. Document ID: US 20050050537 A1

L25: Entry 10 of 21 File: PGPB Mar 3, 2005

PGPUB-DOCUMENT-NUMBER: 20050050537
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050050537 A1

TITLE: Systems and method for representing relationships between units of information manageable by a hardware/software interface system

PUBLICATION-DATE: March 3, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Thompson, J. Patrick	Seattle	WA	US	
Nori, Anil K.	Redmond	WA	US	

US-CL-CURRENT: 717/165; 717/116

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 11. Document ID: US 20050050073 A1

L25: Entry 11 of 21 File: PGPB Mar 3, 2005

PGPUB-DOCUMENT-NUMBER: 20050050073
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050050073 A1

TITLE: Systems and methods for extensions and inheritance for units of information manageable by a hardware/software interface system

PUBLICATION-DATE: March 3, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Demiroski, Bekim	Redmond	WA	US	
Whitney, Robert T.	Seattle	WA	US	
Thompson, J. Patrick	Seattle	WA	US	
Nori, Anil K.	Redmond	WA	US	

US-CL-CURRENT: 707/100

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

---

□ 12. Document ID: US 20050050054 A1

L25: Entry 12 of 21 File: PGPB Mar 3, 2005

PGPUB-DOCUMENT-NUMBER: 20050050054
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050050054 A1

TITLE: Storage platform for organizing, searching, and sharing data

PUBLICATION-DATE: March 3, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Clark, Quentin J.	Bellevue	WA	US	
Nori, Anil K.	Redmond	WA	US	
Celis, Pedro	Redmond	WA	US	
Spiro, Peter M.	Mercer Island	WA	US	
Campbell, David G.	Sammamish	WA	US	

US-CL-CURRENT: 707/100

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

---

□ 13. Document ID: US 20050050053 A1

L25: Entry 13 of 21 File: PGPB Mar 3, 2005

PGPUB-DOCUMENT-NUMBER: 20050050053
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050050053 A1

TITLE: Systems and methods for the implementation of a core schema for providing a top-level structure for organizing units of information manageable by a hardware/software interface system

PUBLICATION-DATE: March 3, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Thompson, J. Patrick	Seattle	WA	US	

US-CL-CURRENT: 707/100

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

---

□ 14. Document ID: US 20050049994 A1

L25: Entry 14 of 21 File: PGPB Mar 3, 2005

PGPUB-DOCUMENT-NUMBER: 20050049994
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050049994 A1

TITLE: Systems and methods for the implementation of a base schema for organizing units of information manageable by a hardware/software interface system

PUBLICATION-DATE: March 3, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Thompson, J. Patrick	Seattle	WA	US	
Cameron, Kim	Bellevue	WA	US	
Smith, Walter R.	Seattle	WA	US	
Nori, Anil K.	Redmond	WA	US	

US-CL-CURRENT: 707/1

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

---

□ 15. Document ID: US 20050049993 A1

L25: Entry 15 of 21 File: PGPB Mar 3, 2005

PGPUB-DOCUMENT-NUMBER: 20050049993
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050049993 A1

TITLE: Systems and methods for data modeling in an item-based storage platform

PUBLICATION-DATE: March 3, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Nori, Anil K.	Redmond	WA	US	
Agarwal, Sameet	Redmond	WA	US	
Thompson, J. Patrick	Seattle	WA	US	
Celis, Pedro	Redmond	WA	US	
Campbell, David G.	Sammamish	WA	US	
Terek, F. Soner	Bellevue	WA	US	

US-CL-CURRENT: 707/1

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

---

□ 16. Document ID: US 20050044530 A1

L25: Entry 16 of 21 File: PGPB Feb 24, 2005

PGPUB-DOCUMENT-NUMBER: 20050044530
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050044530 A1

TITLE: Systems and methods for providing relational and hierarchical synchronization services for units of information manageable by a hardware/software interface system

PUBLICATION-DATE: February 24, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Novik, Lev	Bellevue	WA	US	

US-CL-CURRENT: 717/122; 717/170

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

---

□ 17. Document ID: US 20050044197 A1

L25: Entry 17 of 21 File: PGPB Feb 24, 2005

PGPUB-DOCUMENT-NUMBER: 20050044197
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050044197 A1

TITLE: Structured methodology and design patterns for web services

PUBLICATION-DATE: February 24, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Lai, Ray Y.	Fremont	CA	US	

US-CL-CURRENT: 709/223

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

---

□ 18. Document ID: US 20050044187 A1

L25: Entry 18 of 21 File: PGPB Feb 24, 2005

PGPUB-DOCUMENT-NUMBER: 20050044187
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050044187 A1

TITLE: Systems and methods for providing conflict handling for peer-to-peer synchronization of units of information manageable by a hardware/software interface system

PUBLICATION-DATE: February 24, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Jhaveri, Vivek Jawahir	Seattle	WA	US	
Shah, Ashish	Sammamish	WA	US	
Hudis, Irena	Bellevue	WA	US	
Novik, Lev	Bellevue	WA	US	

US-CL-CURRENT: 709/219

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 19. Document ID: US 20050044108 A1

L25: Entry 19 of 21 File: PGPB Feb 24, 2005

PGPUB-DOCUMENT-NUMBER: 20050044108
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050044108 A1

TITLE: Systems and methods for providing synchronization services for units of information manageable by a hardware/software interface system

PUBLICATION-DATE: February 24, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Shah, Ashish	Sammamish	WA	US	
Shah, Darshatkumar	Bellevue	WA	US	
Hudis, Irena	Bellevue	WA	US	
Novjk, Lev	Bellevue	WA	US	
Jhaveri, Vivek Jawahir	Seattle	WA	US	

US-CL-CURRENT: 707/104.1

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

□ 20. Document ID: US 20050044089 A1

L25: Entry 20 of 21 File: PGPB Feb 24, 2005

PGPUB-DOCUMENT-NUMBER: 20050044089
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050044089 A1

TITLE: Systems and methods for interfacing application programs with an item-based storage platform

PUBLICATION-DATE: February 24, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Wu, Winnie C.	Bellevue	WA	US	
Deem, Michael E.	Redmond	WA	US	
Sheppard, Edward G.	Mercer Island	WA	US	
Fang, Lijiang	Sammamish	WA	US	
Li, Jian	Bellevue	WA	US	
Taylor, Michael B.	Seattle	WA	US	

US-CL-CURRENT: 707/100

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

□ 21. Document ID: US 20030208493 A1

L25: Entry 21 of 21 File: PGPB Nov 6, 2003

PGPUB-DOCUMENT-NUMBER: 20030208493
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030208493 A1

TITLE: Object relational database management system

PUBLICATION-DATE: November 6, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Hall, Bradley S.	Concord	CA	US	
Lunetta, Mark T.	Berkeley	CA	US	

US-CL-CURRENT: 707/100

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Term	Documents
EXTENSION	1035370
EXTENSIONS	263577
(24 AND EXTENSION).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	21
(L24 AND EXTENSION).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	21

Display Format: - Change Format

Previous Page Next Page Go to Doc#

Freeform Search

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L24 and extension

Display: 50 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ **Hit List** ● **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Tuesday, June 21, 2005 Printable Copy Create Case

Set Name **side by side**	**Query**	**Hit Count**	**Set Name** **result set**
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L25	L24 and extension	21	L25
L24	L23 and (relational near database)	21	L24
L23	L22 and (database near engine)	22	L23
L22	L21 and (application near program)	28	L22
L21	L20 and search$	30	L21
L20	L19 and (type near data)	31	L20
L19	L18 and schema$	35	L19
L18	(storage near platform) and (data near model)	41	L18
L17	l13 and (schema$)	3	L17
L16	l13 and (set near schema$)	0	L16
L15	L14 and schema	0	L15
L14	L13 and interface	7	L14
L13	l9 and (storage near platform)	12	L13
DB=USPT; PLUR=YES; OP=OR			
L12	L10 and (data near model)	1	L12
L11	L10 and schema$	3	L11
L10	L9 and (storage near platform)	12	L10
L9	707/$.ccls.	14928	L9

L8	6128610.pn.	1	L8
L7	6370522.pn.	1	L7
L6	6286015.pn.	1	L6
L5	6519647.pn.	1	L5
L4	6134558.pn.	1	L4
L3	6128610.pn.	1	L3
L2	6370522.pn.	1	L2
L1	6738782.pn.	1	L1

END OF SEARCH HISTORY